

PCT/JP2004/012246

日 本 国 特 許 庁
JAPAN PATENT OFFICE

19.08.2004

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日 2 0 0 3 年 8 月 2 9 日
Date of Application:

出 願 番 号 特 願 2 0 0 3 - 3 0 7 5 0 9
Application Number: .
[ST. 10/C]: [J P 2 0 0 3 - 3 0 7 5 0 9]

REC'D 07 OCT 2004	
WIPO	PCT

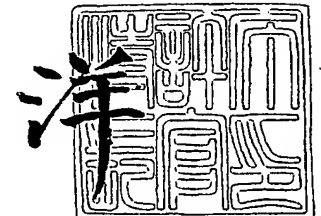
出 願 人 松下電器産業株式会社
Applicant(s):

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

2 0 0 4 年 9 月 2 4 日

特許庁長官
Commissioner,
Japan Patent Office

小 川



【書類名】 特許願
【整理番号】 2038150040
【提出日】 平成15年 8月29日
【あて先】 特許庁長官殿
【国際特許分類】 H04N 4/64
【発明者】
 【住所又は居所】 大阪府門真市大字門真1006番地 松下電器産業株式会社内
 【氏名】 松田 秀治
【発明者】
 【住所又は居所】 大阪府門真市大字門真1006番地 松下電器産業株式会社内
 【氏名】 藪野 寛之
【特許出願人】
 【識別番号】 000005821
 【氏名又は名称】 松下電器産業株式会社
【代理人】
 【識別番号】 100081813
 【弁理士】
 【氏名又は名称】 早瀬 憲一
 【電話番号】 06(6395)3251
【手数料の表示】
 【予納台帳番号】 013527
 【納付金額】 21,000円
【提出物件の目録】
 【物件名】 特許請求の範囲 1
 【物件名】 明細書 1
 【物件名】 図面 1
 【物件名】 要約書 1
 【包括委任状番号】 9600402

【書類名】 特許請求の範囲**【請求項 1】**

それぞれマトリクス状のデータからなる複数のセクタにより構成される対象符号列に対してシンドローム演算を行うシンドローム演算器を有し、該対象符号列に対して誤り訂正を行う誤り訂正回路による誤り訂正処理と同時に、該対象符号列に対して、前記セクタ単位で誤り検出を行う誤り検出装置であって、

前記対象符号列の誤り検出符号を演算する誤り検出符号演算回路と、

前記対象符号列が論理的に連続性のない並びで入力された際に、該符号列に連続性をもたせるスキップ演算を行う誤り検出符号スキップ演算回路と、

前記シンドローム演算と同じ時点で行う第 1 の誤り検出符号演算処理を制御する第 1 の誤り検出制御回路と、

前記誤り訂正処理後に、該誤り訂正処理より得られた誤りデータ位置及び誤りデータ数値を元に、該誤りデータ位置が示すデータに対してのみ行う第 2 の誤り検出符号演算処理を制御すると共に、該第 2 の誤り検出符号演算処理による演算結果を元に、前記第 1 の誤り検出符号演算処理の演算結果を更新する更新処理を制御する第 2 の誤り検出制御回路と

、
前記誤り検出符号演算回路、及び前記誤り検出符号スキップ演算回路による演算結果を保持するメモリと、を備える、

ことを特徴とする誤り検出装置。

【請求項 2】

請求項 1 に記載の誤り検出装置において、

前記誤り検出符号スキップ演算回路は、前回までに入力された対象符号列の誤り検出符号を入力とし、該誤り検出符号スキップ演算回路に予め設定されたスキップ演算を行う、

ことを特徴とする誤り検出装置。

【請求項 3】

請求項 1 に記載の誤り検出装置において、

前記メモリは、前記対象符号列が連続的な並びで入力された際は、前記誤り検出符号演算回路、及び前記誤り検出符号スキップ演算回路によるセクタ単位の演算結果を保持し、前記対象符号列が連続的でない並びで入力された際は、前記誤り検出符号演算回路、及び前記誤り検出符号スキップ演算回路によるセクタ毎の途中演算結果を保持する第 1 のメモリと、

前記第 1 のメモリから送信される演算結果を、前記各セクタ毎に保持する第 2 のメモリと、を備える、

ことを特徴とする誤り検出装置。

【請求項 4】

請求項 1 に記載の誤り検出装置において、

前記第 1 のメモリは、前記誤り検出符号演算回路、及び前記誤り検出符号スキップ演算回路による演算結果を保持する、少なくとも 2 つの誤り検出符号演算結果保持部からなる

、
ことを特徴とする誤り検出装置。

【請求項 5】

請求項 4 に記載の誤り検出装置において、

前記少なくとも 2 つの誤り検出符号演算結果保持部のうちの 1 つは、前記第 1 の誤り検出符号演算処理による演算結果を保持する、

ことを特徴とする誤り検出装置。

【請求項 6】

請求項 4 に記載の誤り検出装置において、

前記少なくとも 2 つの誤り検出符号演算結果保持部のうちの 1 つは、前記第 2 の誤り検出符号演算処理による、前記第 1 の誤り検出符号演算処理の演算結果を更新する差分演算結果を保持する、

ことを特徴とする誤り検出装置。

【請求項 7】

請求項 1 に記載の誤り検出装置において、

前記誤り検出符号スキップ演算回路は、前記対象符号列が論理的に連続性のない並びで入力されたとき、該対象符号列のうち前記セクタの非最終行においては、特定のスキップ演算を行い、前記対象符号列のうち前記セクタの最終行においては、該データが存在する列位置に応じた個別のスキップ演算を行う、

ことを特徴とする誤り検出装置。

【請求項 8】

それぞれマトリクス状のデータからなる複数のセクタにより構成される、論理的に連続性のない並びで入力された対象符号列に対して誤り訂正を行うと同時に、該連続性のない並びで入力される対象符号列に対して前記セクタ単位で誤り検出を行う誤り検出方法であって、

前記対象符号列に対してシンドローム演算を行うシンドローム演算ステップと、

前記シンドローム演算ステップと同時に行われる、該連続性のない並びで入力される対象符号列に対して誤り検出符号演算を行う第 1 の誤り検出符号演算ステップと、

前記シンドローム演算ステップにおいて得られたシンドロームに基づいて、前記対象符号列の誤りデータ位置及び誤りデータ数値を計算して誤り訂正を行う誤り訂正ステップと

、
前記誤り訂正ステップにおいて得られた前記誤りデータ位置及び誤りデータ数値を元に、前記対象符号列のうちの前記誤りデータ位置に対してのみ再度誤り検出符号演算を行う第 2 の誤り検出符号演算ステップと、

前記第 2 の誤り検出符号演算ステップによる演算結果を用いて、前記第 1 の誤り検出符号演算ステップによる演算結果を更新する更新ステップと、を含み、

前記第 1 の誤り検出符号演算ステップ及び第 2 の誤り検出符号演算ステップは、前記対象符号列の誤り検出符号を演算する誤り検出符号演算ステップと、前記論理的に連続性のない並びで入力された対象符号列に連続性をもたせるスキップ演算を行う誤り検出符号スキップ演算ステップと、を含む、

ことを特徴とする誤り検出方法。

【請求項 9】

請求項 8 に記載の誤り検出方法において、

前記誤り検出符号スキップ演算ステップは、前回までに入力された対象符号列の誤り検出符号を入力とし、予め設定されたスキップ演算を行う、

ことを特徴とする誤り検出方法。

【請求項 10】

請求項 8 に記載の誤り検出方法において、

前記誤り検出符号スキップ演算ステップは、前記対象符号列のうち前記セクタの非最終行においては、特定のスキップ演算を行い、前記対象符号列のうち前記セクタの最終行においては、該データが存在する列位置に応じた個別のスキップ演算を行う、

ことを特徴とする誤り検出方法。

【請求項 11】

請求項 10 に記載の誤り検出方法において、

前記個別のスキップ演算は、前記データが存在する列位置のうち、特定の列位置で実行されるステップ演算の演算結果を複数回利用して行う、

ことを特徴とする誤り検出方法。

【請求項 12】

請求項 8 に記載の誤り検出方法において、

前記更新ステップと同時に行われる、前記対象符号列に含まれるスクランブル成分を除去するスクランブル除去ステップ、または、該対象符号列にスクランブル成分を付加するスクランブル付加ステップを含む、

ことを特徴とする誤り検出方法。

【請求項 13】

請求項 12 に記載の誤り検出方法において、
前記スクランブル除去ステップは、1 セクタの全データが入力された後に、該セクタの全データのスクランブル成分を一度に除去する、
ことを特徴とする誤り検出方法。

【請求項 14】

請求項 12 に記載の誤り検出方法において、
前記スクランブル除去ステップは、前記対象符号列のスクランブル成分を除去するためのデータを保持するテーブルを用いて行われる、
ことを特徴とする誤り検出方法。

【請求項 15】

請求項 8 に記載の誤り検出方法において、
前記誤り訂正及び誤り検出を行う対象データは、光記録媒体に記録されているデータである、
ことを特徴とする誤り検出方法。

【書類名】明細書

【発明の名称】誤り検出装置及び誤り検出方法

【技術分野】

【0001】

本発明は、誤り検出装置及び誤り検出方法に関し、特に誤り訂正と誤り検出を同時に実行するための誤り検出装置及び誤り検出方法に関する。

【背景技術】

【0002】

デジタルデータの記録／再生を行うシステムでは、再生時もしくは記録時にデータ中に誤りが発生することがあることから、その誤りを訂正し、行った誤り訂正が正しいものかどうかを判定する誤り検出を行う必要がある。

【0003】

そして、DVDの記録／再生においては、誤り訂正ではECC (Error Correcting Code)、誤り検出ではEDC (Error Detecting Code) と呼ばれる方法が用いられる。この誤り訂正はECCブロック単位、誤り検出はセクタ単位で実行される。

【0004】

まず、本発明にかかる誤り検出装置の理解を容易にするために、図9～図12を用いて、ECC復号化の概要について説明する。図9は、DVD上のECCブロックの構成を示す図であり、図10は、図9のECCブロックのC1方向についての誤り訂正を説明する図であり、図11は、図9のECCブロックのC2方向についての誤り訂正を説明する図であり、図12は、DVD上のセクタの構成を示す図である。

【0005】

なお、ここでは、光記録媒体であるDVDに記録されているデータを誤り訂正する場合を例に挙げて説明するものとし、誤り訂正を行う一単位であるECCブロックは、図9に示すように16のセクタより構成され、その1セクタは、図12に示すような構成を有するものとする。

【0006】

まず、DVDに記録されているデータを誤り訂正する場合、そのDVDから読み出したECC符号化されたデータを復号し、図9に示すC1方向もしくはC2方向について誤り訂正を行う。このとき、ECC復号したデータから位置多項式及び数値多項式を生成し、それらの根を求めることにより誤りデータ位置及び誤りデータ数値を算出する。

【0007】

このC1方向あるいはC2方向について誤り訂正を行うときに、誤り訂正能力を超える誤りが存在した場合は、図10に示すように、その誤り訂正能力を超える符号列を訂正不能符号列とし、この訂正不能符号列に関する情報を消失位置情報として記憶しておく。

【0008】

そして、1 ECCブロック中の全ての符号列に対する、C1方向もしくはC2方向についての誤り訂正が完了した後、前回と異なる方向 (C2方向もしくはC1方向) について、前記消失位置情報を用いて誤り訂正を行う。

【0009】

このように、誤り訂正を行う際に予め誤りデータの位置が分かっている場合には、該誤りデータの位置を示す消失位置情報を利用することで、前記位置多項式及び数値多項式を生成する際に、数値多項式のみを求めれば良いことになる。その結果、誤り訂正能力を向上させることができる。

【0010】

例えば、図10に示すように、最初に1 ECCブロック中の全ての符号列に対してC1方向について誤り訂正を行い、50、90、130、200行目が訂正不能符号列であったとする。この場合、図11に示すように、次の誤り訂正方向であるC2方向の誤り訂正を行うときに、前回の訂正不能符号列を示す消失位置情報をもとに50、90、130、200Byte目を消失位置情報と指定することで、C2方向の誤り訂正能力を向上させることができる。

【0011】

以上の方法により、伝送系で発生したバースト誤りに対して訂正能力を落とすことなく誤り訂正が可能となる。

【0012】

そして、このようなECC処理の後、従来においては、前述のようにしてECC処理が施されたデータに対して、セクタ単位でEDC演算を行っていく。

【0013】

以下、図12及び図13を用いて、EDC演算の概要について説明する。図13はEDC演算回路の構成例を示す図であり、ここでは、4 Byte入力のEDC演算回路を示す。

【0014】

以下、EDCの演算式を示す。

【数1】

$$\text{EDC}(x) = \sum_{i=31}^0 b_i x^i = l(x) \bmod \{g(x)\}$$

$$l(x) = \sum_{i=16511}^{32} b_i x^i \quad g(x) = x^{32} + x^{31} + x^4 + 1$$

【0015】

前記EDC演算式は、図13のEDC演算回路に示されるように、32ビットのシフトレジスタで、0ビット目、4ビット目、31ビット目への入力にEXOR演算回路が配置され、それぞれ、4 Byteの最上位からのビット入力と31ビット目、3ビット目と31ビット目、及び30ビット目と31ビット目のEXORが演算される。

【0016】

以上に示したEDC演算式及びEDC演算回路から理解できるように、EDC演算はデータの記録方向（図9に示されるC1方向）に沿った線形演算である。そして、図12に示すセクタを、データの記録方向に沿ってID領域、IEC領域、RSV領域、ユーザデータ領域、EDC領域の順序で、それぞれ4 Byte毎に図13に示すEDC演算回路に入力すると、最終の4 ByteであるEDC領域を入力した後の32ビットのシフトレジスタ値がEDC演算結果となる。

【特許文献1】特開平7-230388号公報

【特許文献2】特開平5-315974号公報

【特許文献3】特開平7-50595号公報

【特許文献4】特許第3272317号公報

【発明の開示】

【発明が解決しようとする課題】

【0017】

このように、従来では一般的に、誤り訂正（ECC）を行った後に誤り検出（EDC）を行う方法が用いられているが、この方法では、誤り訂正の一単位であるECCブロックをバッファから読み出して誤り訂正を行った後、再度誤り検出を行うためにECCブロックをバッファから読み出さなければならぬため、メモリバッファのバンド幅の消費、処理時間の肥大化が発生する。

【0018】

このような課題に鑑み、バッファからの一度の読み出しでECCとEDCの同時処理を行うことを考えた場合、前述したようにEDC演算処理は、データの記録方向（図9のC1方向）に沿った線形演算であるため、図10に示すように誤り訂正装置に対して入力されるデータに論理的な連続性があるC1方向ECCの場合は、該C1方向のECCとEDC演算の同時処理が可能であるが、図11に示すように誤り訂正装置に対して入力されるデータに論理的な連続性がないC2方向のECCの場合は、前記EDC演算と前記C2方向のECCとを同時処理するのは半導体集積回路の実装における難度が高いという課題がある。

【0019】

本発明は前記課題を解決するためになされたものであり、ECCブロックの誤り検出を行う対象符号列が論理的に連続的でない並び（図9のC2方向）で入力された際にも、誤り訂正と誤り検出を同時に行える誤り検出装置及び誤り検出方法を提供することを目的とするものである。

【課題を解決するための手段】**【0020】**

本発明の誤り検出装置は、それぞれマトリクス状のデータからなる複数のセクタにより構成される対象符号列に対してシンδροーム演算を行うシンδροーム演算器を有し、該対象符号列に対して誤り訂正を行う誤り訂正回路による誤り訂正処理と同時に、該対象符号列に対して、前記セクタ単位で誤り検出を行う誤り検出装置であって、前記対象符号列の誤り検出符号を演算する誤り検出符号演算回路と、前記対象符号列が論理的に連続性のない並びで入力された際に、該符号列に連続性をもたせるスキップ演算を行う誤り検出符号スキップ演算回路と、前記シンδροーム演算と同じ時点で行う第1の誤り検出符号演算処理を制御する第1の誤り検出制御回路と、前記誤り訂正処理後に、該誤り訂正処理より得られた誤りデータ位置及び誤りデータ数値を元に、該誤りデータ位置が示すデータに対してのみ行う第2の誤り検出符号演算処理を制御すると共に、該第2の誤り検出符号演算処理による演算結果を元に、前記第1の誤り検出符号演算処理の演算結果を更新する更新処理を制御する第2の誤り検出制御回路と、前記誤り検出符号演算回路、及び前記誤り検出符号スキップ演算回路による演算結果を保持するメモリと、を備えるものである。

【0021】

これにより、ECCブロックの誤り検出を行う対象符号列が論理的に連続的でない並びで入力された際にも、誤り訂正と誤り検出を同時に行える。

【0022】

さらに、本発明の誤り検出装置は、前記誤り検出符号スキップ演算回路は、前回までに入力された対象符号列の誤り検出符号を入力とし、該誤り検出符号スキップ演算回路に予め設定されたスキップ演算を行うものである。

【0023】

これにより、簡単な処理でスキップ演算が行え、誤り訂正と誤り検出の同時実行を、半導体回路に実装可能にする。

【0024】

さらに、本発明の誤り検出装置は、前記メモリは、前記対象符号列が連続的な並びで入力された際は、前記誤り検出符号演算回路、及び前記誤り検出符号スキップ演算回路によるセクタ単位の演算結果を保持し、前記対象符号列が連続的でない並びで入力された際は、前記誤り検出符号演算回路、及び前記誤り検出符号スキップ演算回路によるセクタ毎の途中演算結果を保持する第1のメモリと、前記第1のメモリから送信される演算結果を、前記各セクタ毎に保持する第2のメモリ、とを備えるものである。

【0025】

これにより、ECCブロックの誤り検出を行う対象符号列が論理的に連続的でない並びで入力された際にも、誤り訂正と誤り検出とを同時に行うことが可能となる。

【0026】

さらに、本発明の誤り検出装置は、前記第1のメモリは、前記誤り検出符号演算回路、及び前記誤り検出符号スキップ演算回路による演算結果を保持する、少なくとも2つの誤り検出符号演算結果保持部からなるものである。

【0027】

さらに、本発明の誤り検出装置は、前記少なくとも2つの誤り検出符号演算結果保持部のうちの1つは、前記第1の誤り検出符号演算処理による演算結果を保持するものである。

【0028】

さらに、本発明の誤り検出装置は、前記少なくとも2つの誤り検出符号演算結果保持部

のうちの1つは、前記第2の誤り検出符号演算処理による、前記第1の誤り検出符号演算処理の演算結果を更新する差分演算結果を保持するものである。

【0029】

これにより、前回入力されたデータと、次に入力されたデータ間にデータの論理的な連続性をもたない場合においても、EDC演算処理を行うことが可能となる。

【0030】

さらに、本発明の誤り検出装置は、前記誤り検出符号スキップ演算回路は、前記対象符号列が論理的に連続性のない並びで入力されたとき、該対象符号列のうち前記セクタの非最終行においては、特定のスキップ演算を行い、前記対象符号列のうち前記セクタの最終行においては、該データが存在する列位置に応じた個別のスキップ演算を行うものである。

【0031】

これにより、簡単な処理でスキップ演算が行え、誤り訂正と誤り検出の同時実行を、半導体回路に実装可能にする。

【0032】

また、本発明の誤り検出方法は、それぞれマトリクス状のデータからなる複数のセクタにより構成される、論理的に連続性のない並びで入力された対象符号列に対して誤り訂正を行うと同時に、該連続性のない並びで入力される対象符号列に対して前記セクタ単位で誤り検出を行う誤り検出方法であって、前記対象符号列に対してシンドローム演算を行うシンドローム演算ステップと、前記シンドローム演算ステップと同時に行われる、該連続性のない並びで入力される対象符号列に対して誤り検出符号演算を行う第1の誤り検出符号演算ステップと、前記シンドローム演算ステップにおいて得られたシンドロームに基づいて、前記対象符号列の誤りデータ位置及び誤りデータ数値を計算して誤り訂正を行う誤り訂正ステップと、前記誤り訂正ステップにおいて得られた前記誤りデータ位置及び誤りデータ数値を元に、前記対象符号列のうちの前記誤りデータ位置に対してのみ再度誤り検出符号演算を行う第2の誤り検出符号演算ステップと、前記第2の誤り検出符号演算ステップによる演算結果を用いて、前記第1の誤り検出符号演算ステップによる演算結果を更新する更新ステップと、を含み、前記第1の誤り検出符号演算ステップ及び第2の誤り検出符号演算ステップは、前記対象符号列の誤り検出符号を演算する誤り検出符号演算ステップと、前記論理的に連続性のない並びで入力された対象符号列に連続性をもたせるスキップ演算を行う誤り検出符号スキップ演算ステップと、を含むものである。

【0033】

これにより、ECCブロックの誤り検出を行う対象符号列が論理的に連続的でない並びで入力された際にも、誤り訂正と誤り検出を同時に行える。

【0034】

さらに、本発明の誤り検出方法は、前記誤り検出符号スキップ演算ステップは、前回までに入力された対象符号列の誤り検出符号を入力とし、予め設定されたスキップ演算を行うものである。

【0035】

これにより、簡単な処理でスキップ演算が行え、入力されるデータが論理的に連続性のない場合でも、誤り訂正と誤り検出との同時実行を、容易に実現することができる。

【0036】

さらに、本発明の誤り検出方法は、前記誤り検出符号スキップ演算ステップは、前記対象符号列のうち前記セクタの非最終行においては、特定のスキップ演算を行い、前記対象符号列のうち前記セクタの最終行においては、該データが存在する列位置に応じた個別のスキップ演算を行うものである。

【0037】

これにより、誤り訂正と誤り検出の同時実行を、容易に実現することができる。

【0038】

さらに、本発明の誤り検出方法は、前記個別のスキップ演算は、前記データが存在する

列位置のうち、特定の列位置で実行されるステップ演算の演算結果を複数回利用して行うものである。

【0039】

これにより、セクタ内の誤り検出演算処理単位列に対応した、全てのスキップ演算処理を用意することなく、別個に存在するスキップ演算を複数回再利用することができ、この結果、処理資源の消費を抑えることが可能となる。

【0040】

さらに、本発明の誤り検出方法は、前記更新ステップと同時に行われる、前記対象符号列に含まれるスクランブル成分を除去するスクランブル除去ステップ、または、該対象符号列にスクランブル成分を付加するスクランブル付加ステップを含むものである。

【0041】

これにより、ECC処理及びEDC処理後のデータから、スクランブル成分を除去でき、あるいは該処理後のデータにスクランブル成分を付加できる。

【0042】

さらに、本発明の誤り検出方法は、前記スクランブル除去ステップは、1セクタの全データが入力された後に、該セクタの全データのスクランブル成分を一度に除去するものである。

【0043】

これにより、より簡単な処理によって、ECC処理及びEDC処理後のデータから、スクランブル成分を除去でき、処理時間を削減できる。

【0044】

さらに、本発明の誤り検出方法は、前記スクランブル除去ステップは、前記対象符号列のスクランブル成分を除去するためのデータを保持するテーブルを用いて行われるものである。

【0045】

これにより、ECC処理及びEDC処理後のデータから、全データのスクランブル成分を一度に除去することができる。

【0046】

さらに、本発明の誤り検出方法は、前記誤り訂正及び誤り検出を行う対象データは、光記録媒体に記録されているデータであるものである。

【発明の効果】

【0047】

本発明の誤り検出装置によれば、それぞれマトリクス状のデータからなる複数のセクタにより構成される対象符号列に対してシンドローム演算を行うシンドローム演算器を有し、該対象符号列に対して誤り訂正を行う誤り訂正回路による誤り訂正処理と同時に、該対象符号列に対して、前記セクタ単位で誤り検出を行う誤り検出装置であって、前記対象符号列の誤り検出符号を演算する誤り検出符号演算回路と、前記対象符号列が論理的に連続性のない並びで入力された際に、該符号列に連続性をもたせるスキップ演算を行う誤り検出符号スキップ演算回路と、前記シンドローム演算と同じ時点で行う第1の誤り検出符号演算処理を制御する第1の誤り検出制御回路と、前記誤り訂正処理後に、該誤り訂正処理より得られた誤りデータ位置及び誤りデータ数値を元に、該誤りデータ位置が示すデータに対してのみ行う第2の誤り検出符号演算処理を制御すると共に、該第2の誤り検出符号演算処理による演算結果を元に、前記第1の誤り検出符号演算処理の演算結果を更新する更新処理を制御する第2の誤り検出制御回路と、前記誤り検出符号演算回路、及び前記誤り検出符号スキップ演算回路による演算結果を保持するメモリと、を備えるようにしたので、EDC演算処理を行う際に、入力されるデータが論理的な連続性のない方向であっても、EDC演算処理を実行することが可能となる。また、これにより、ECC処理とEDC処理を同時実施が可能となるので、ECCブロックをバッファから一度読み出すだけで両処理が行え、ECC処理のためにバッファから読み出した1ECCブロックを、ECC処理後に再度読み出してEDC演算処理を行う必要がなくなり、メモリバッファのバンド幅の消費、及び処理時の肥

大化を低減することができる。

【0048】

また、本発明の誤り検出方法によれば、それぞれマトリクス状のデータからなる複数のセクタにより構成される、論理的に連続性のない並びで入力された対象符号列に対して誤り訂正を行うと同時に、該連続性のない並びで入力される対象符号列に対して前記セクタ単位で誤り検出を行う誤り検出方法であって、前記対象符号列に対してシンドローム演算を行うシンドローム演算ステップと、前記シンドローム演算ステップと同時に行われる、該連続性のない並びで入力される対象符号列に対して誤り検出符号演算を行う第1の誤り検出符号演算ステップと、前記シンドローム演算ステップにおいて得られたシンドロームに基づいて、前記対象符号列の誤りデータ位置及び誤りデータ数値を計算して誤り訂正を行う誤り訂正ステップと、前記誤り訂正ステップにおいて得られた前記誤りデータ位置及び誤りデータ数値を元に、前記対象符号列のうちの前記誤りデータ位置に対してのみ再度誤り検出符号演算を行う第2の誤り検出符号演算ステップと、前記第2の誤り検出符号演算ステップによる演算結果を用いて、前記第1の誤り検出符号演算ステップによる演算結果を更新する更新ステップと、を含み、前記第1の誤り検出符号演算ステップ及び第2の誤り検出符号演算ステップは、前記対象符号列の誤り検出符号を演算する誤り検出符号演算ステップと、前記論理的に連続性のない並びで入力された対象符号列に連続性をもたせるスキップ演算を行う誤り検出符号スキップ演算ステップと、を含むようにしたので、入力されるデータが論理的な連続性のない方向のECCと同時にEDC演算処理を行うことができ、ECCブロックをバッファから一度読み出すだけで両処理が行え、メモリバッファのバンド幅の消費、及び処理時の肥大化を低減することができる。

【発明を実施するための最良の形態】

【0049】

以下、図1～図6を用いて、本実施の形態1に係る、誤り訂正と誤り検出を同時処理する誤り検出装置、及び誤り検出方法について説明する。なお、本実施の形態1においては、光記録媒体であるDVDに記録されているデータを誤り訂正(ECC)及び誤り検出(EDC)する場合を例に挙げて説明するものとし、誤り訂正を行う一単位であるECCブロックは、図9に示すように16のセクタより構成され、その1セクタは、図12に示すような構成を有するものとする。

【0050】

まず、図1を用いて、本実施の形態1にかかる、ECC処理とEDC処理の同時実行処理、及び入力されるデータ符号列のスクランブル成分除去について説明する。図1は、本実施の形態1における、ECC処理とEDC処理の同時実行におけるパイプライン処理を説明する図である。

【0051】

まずステップS10において、DVDから、誤り訂正及び誤り検出を行うべきデータ符号列をC1方向あるいはC2方向の順に入力して、ECC処理のシンドローム演算処理を行う。そして、前記ステップS10と同じパイプライン処理ステージであるステップS20において、前記ステップS10と同じ順番で入力されるデータ符号列に対して第1のEDC演算処理を実行し、セクタ単位毎に第1の誤り検出符号演算結果を得る。この第1の誤り検出符号演算結果は、入力されたデータ符号列に誤りが含まれていない場合、正常な値となり、誤りが含まれているとき、不正な値となる。

【0052】

そして、前記ステップS10により得られるシンドローム演算結果が0となり、入力されたデータ符号列に誤りが存在しないと判定された場合、ステップS20以降の誤り訂正処理は行われずに、処理を終了する。そしてこの場合、前記ステップS20において得られた第1の誤り検出符号演算結果が、当該誤り検出装置10より得られる正しいEDC演算結果として出力される。

【0053】

一方、ステップS10により得られるシンドローム演算結果が0とならず、入力された

データ符号列に誤りが存在すると判定された場合は、引き続き、前記誤り訂正回路20において誤り訂正処理が行われ、入力された符号列より誤りデータ位置及び誤りデータ数値が得られる(ステップS30)。

【0054】

そして、ステップS40においては、前記ステップS30で得られた誤りデータ位置、及び誤りデータ数値を元にして、DVDから入力されるデータ符号列のうち、該得られた誤り位置が示すデータに対してのみ、第2のEDC演算処理を実行して、第2の誤り検出符号演算結果を得る。このときに得られる第2の誤り検出符号演算結果は、当該誤り検出装置10より得られる正しいEDC演算結果と、前記ステップS20において得られた前記第1の誤り検出符号演算結果との差分情報である。

【0055】

よって、ステップS50において、前記ステップS40で得られた第2の誤り検出符号演算結果を用いて、前記ステップS20において得られた第1の誤り検出符号演算結果を更新し、その更新後の誤り検出符号演算結果が、当該誤り検出装置10より得られる正しいEDC演算結果として出力される。

【0056】

そして、ステップS50と同じパイプライン処理ステージであるステップS60で、スクランブル成分を除去するためのデータを保持するテーブルを用いて、入力されるデータ符号列のスクランブル成分を一括で除去する処理が行われる。

【0057】

以上のような一連の処理を行うことにより、ECC処理とEDC処理の同時実行、そして入力されるデータ符号列のスクランブル成分除去を、ECCブロックをバッファから1度だけ読み出すことによって、実行することが可能となる。

【0058】

なお、前述の説明では、DVDに記録されているデータを再生する場合を例に挙げているため、ECC処理と同時にEDC演算した後、前記データ符号列のスクランブル成分を除去するものとして説明したが、DVDにデータを記録する場合は、記録するデータをECC処理すると同時にEDC演算した後、前記データのスクランブル成分を付加することとなる。ただし、スクランブル成分を付加する場合は、ステップS60に示すスクランブル除去のように、一括して行うことはできないため、EDC演算されたデータが出力される度に付加していく。

【0059】

次に、図2を用いて、上述したようなECC処理とEDC処理の同時処理が実行可能な、本実施の形態1における誤り検出装置の構成について説明する。図2は、本実施の形態1にかかる誤り検出装置の構成を示す図である。

【0060】

本実施の形態1に係る誤り検出回路10は、前記データ符号列に対してシンドローム演算を行うシンドローム演算器21を有する誤り訂正回路20がECC処理を実施すると同時に、EDC処理を実施可能なものであって、図1に示すように、誤り検出符号演算回路111と誤り検出符号スキップ演算回路112とからなる演算回路11と、第1の誤り検出符号演算結果保持部131と第2の誤り検出符号演算結果保持部132とからなる第1のメモリ13と、第2のメモリ12と、第1の誤り検出制御回路141と第2の誤り検出制御回路142とからなる制御回路14と、で構成されるものである。

【0061】

以下、各回路について説明すると、前記制御回路14は、当該誤り検出回路10に入力されるデータ符号列に対して誤り検出を行う際に、当該誤り検出回路10を制御するものであり、前記第1の誤り検出制御回路141は、前記誤り訂正回路20内のシンドローム演算器21における前記データ符号列に対するシンドローム演算と同時に実行される第1のEDC演算処理(図1のステップS20)を制御し、前記第2の誤り検出制御回路142は、前記誤り訂正回路20による誤り訂正処理後の第2のEDC演算処理、及び前記第1のEDC演算結果の更新処理(図1のステップS40, S50)を制御するものである。

【0062】

そして、前記演算回路11は、前記制御回路14の制御の下、入力される前記データ符号列に対して誤り検出演算を行うものであり、前記誤り検出符号演算回路111は、前記DVDから入力されるデータ符号列の誤り検出符号を任意のデータ毎に（ここでは4バイト毎）検出するものであり、前記誤り検出符号スキップ演算回路112は、図9のC2方向に前記データ符号列が入力された場合に、データに連続性をもたせるためのスキップ演算を行うものである。

【0063】

そして、前記第1のメモリ13は、前記演算回路11において得られた途中演算結果を保持していくものであり、前記第1の誤り検出符号演算結果保持部131は、前記第1の誤り検出制御回路141の制御の下で行われる第1のEDC演算（図1のステップS20）で得られた演算結果を保持するものであり、前記第2の誤り検出符号演算結果保持部132は、前記第2の誤り検出制御回路142の制御の下で行われる第2のEDC演算（図1のステップS40）で得られる演算結果、つまり前記第1のEDC演算処理による演算結果を更新するための差分情報を保持するものである。

【0064】

そして、前記第2のメモリ12は、前記第1のメモリ13から出力されるデータをセクタ毎に保持するものであり、前記第2のEDC演算処理後には、前記第2のEDC演算結果により第1のEDC演算結果が更新され、該第2のメモリ12には、正しいEDC演算結果が保持される。

【0065】

以上のような構成を有する本実施の形態1における誤り検出装置10により、ECC処理とEDC処理の同時実行処理の一連の動作について、図3～図6を用いて説明する。図3は、本実施の形態1に係る、C2方向訂正時に第0セクタにおいてデータ入力単位を4Byteとしたときの誤り訂正回路、及び誤り検出回路に対するデータ入力順序を示す図であり、図4は、本実施の形態1における誤り検出回路において、C2方向訂正時の第1のEDC演算処理の一連の流れを示すフローチャート図であり、図5は、C2方向訂正時における、第1のEDC演算処理時のデータの流れを示す図であり、図6は、C2方向訂正時における、第2のEDC演算処理時のデータの流れを示す図である。なお、前述と同様、誤り訂正回路20及び誤り検出回路10に入力される1ECCブロックは図9に示されるような構成を有し、またセクタは図12に示される構成を有するものとして説明する。

【0066】

以下、図9におけるC1方向あるいはC2方向のECC処理のうち、誤り検出を行う対象符号列が連続的でないC2方向の並びでECC処理が行われる場合における、誤り検出回路10による一連のEDC処理動作について説明する。

【0067】

まず、誤り訂正回路20及び誤り検出回路10に対して入力されるデータ符号列は、図3に示すようにECCブロックのC2方向に沿って入力される。ここでは、データの入力単位及びEDC演算処理の単位が4Byteであるとする。

【0068】

そして、この入力されたデータ符号列は、誤り訂正回路20内のシンドローム演算器21にて、シンドローム演算が実行されると同時に（図1のステップS10）、本誤り検出回路10内の演算回路11にて、第1の誤り検出制御回路141の制御の下、第1のEDC演算処理が行われる。以下、その詳細な処理について、図4及び図5を参照しながら説明する。なお、図9に示す1ECCブロックには、16のセクタ（第1セクタ～第15セクタ）が含まれているが、各セクタにおいて処理方法は同様であり、ここでは説明を簡略化するために、16セクタのうち第0セクタのみを例に挙げて説明する。

【0069】

図4に示すように、第1のEDC演算処理が開始されると、第1の誤り検出制御回路141内に保持されているセクタ内行数カウンタ“ROW”と、セクタ内列数カウンタ“COL”を

初期化する（ステップS201,S202）。そして、第1のメモリ13内の第1誤り検出符号演算結果保持部131の値（以下、「TMP_EDC_VAL_1」と称す。）、及び第2のメモリ12の値（以下、「EDC_VAL」と称す。）を初期化する（ステップS203,S204）。

【0070】

この後、誤り検出回路10及び誤り訂正回路20において、前記データ符号列を4Byte受信する。この受信したデータは、図3に示す“0”番目のデータである（ステップS205）。

【0071】

そして、受信したデータに対して、誤り検出符号演算回路111において、後述する4Byte入力のEDC演算処理を行い（ステップS206）、その演算結果と、“TMP_EDC_VAL_1”のEXORをとった値を、再度“TMP_EDC_VAL_1”に保持する（ステップS207）。

【0072】

ここで、“0”番目のデータの演算が終了後、次に入力されるデータ符号列は、図3に示す“1”番目のデータであり、先に入力された“0”番目のデータとの論理的な連続性をもたない。

【0073】

そこで、ステップS208において、現在処理中のセクタ内行数カウンタ“ROW”を判定し、該セクタ内行数カウンタ“ROW”が、セクタ内最終行である“11”でなければ、“0”番目のデータと“1”番目のデータ間の論理的な連続性を補正するために、前記第1の誤り制御回路141の制御の下、前記演算回路11内の誤り検出符号スキップ演算回路112によって、168ByteスキップEDCスキップ演算処理を行う（ステップS209）。この処理の詳細については後述する。そして、この誤り検出符号スキップ演算回路112により得たスキップ演算結果と、前記“TMP_EDC_VAL_1”のEXORをとった値を、再度“TMP_EDC_VAL_1”に保持する（ステップS210）。この後、前記第1の誤り訂正検出回路141は、セクタ内行数カウンタ“ROW”を1インクリメントし（ステップS211）、次行のEDC演算処理に移行する。これらの処理を、前記セクタ内行数カウンタ“ROW”がセクタ内最終行である“11”になるまで繰り返す（ステップS205～S211）。

【0074】

そして、ステップS208において、前記セクタ内行数カウンタ“ROW”が“11”になった場合は、図3に示す“11”番目もしくは“219”番目、・・・、“8539”番目のデータ入力に対応した4Byte入力のEDC演算処理が終了している状態となる。そして、前記セクタ内列数カウンタ“COL”が“42”になるまでは、前記セクタ内行数カウンタ“ROW”が“11”になる度に、前記誤り検出符号スキップ演算回路112においてその次に行うべきEDCスキップ演算処理は、前記第1の誤り検出制御回路141により、現在のセクタ内列数カウンタ“COL”の値に対応した個別の処理となるよう制御される（ステップS213）。

【0075】

例えば、ステップS213の時点で、現在前記セクタ内列数カウンタ“COL”が“1”、つまり、図3に示す“219”番目のデータまでEDC演算処理が終了している状態であれば、該“219”番目のデータと、図3に示すセクタの最終データでありEDC領域である“8747”番目のデータまでの差分は164Byteであるため、前記第1の誤り検出制御回路141は、前記誤り検出符号スキップ演算回路112において、164ByteスキップさせるEDCスキップ演算処理を行うように制御する。また、前記ステップS213の時点で、前記セクタ内列数カウンタ“COL”が“41”、つまり図3に示す“8539”番目のデータまでのEDC演算処理が終了している状態であれば、該“8539”番目のデータと、図3に示す最終データでありEDC領域である“8747”番目のデータまでの差分は4Byteであるため、前記第1の誤り検出制御回路141は、前記誤り検出符号スキップ演算回路112において、4ByteスキップさせるEDCスキップ演算処理を行うように制御する。

【0076】

そして、前記セクタ内列数カウンタ“COL”値毎に異なる、前記誤り検出符号スキップ演算回路112によるスキップ演算結果と、“TMP_EDC_VAL_1”とのEXORをとった値を、

再度“TMP_EDC_VAL_1”に保持する（ステップS214）。

【0077】

この時点で、第1のメモリ13内の、第1の誤り検出符号演算結果保持部131（“TMP_EDC_VAL_1”）には、セクタ内1列分の演算結果が保持されたこととなるので、図5に示すように、第1の誤り制御回路141により、この保持された演算値を第2のメモリ12（“EDC_VAL”）に出力して、該第2のメモリ12内の対応するセクタレジスタに出力して保持する。つまり、“EDC_VAL”と“TMP_EDC_VAL_1”とのEXORをとった値を、再度“EDC_VAL”に保持する（ステップS215）。例えば、現在第0セクタ内の“COL”が“1”、つまり図3に示す“219”番目のデータまでのEDC演算が終了し、“TMP_EDC_VAL_1”には、図3の“208”～“219”番目のデータまでのEXORをとった値が保持されている場合、図5に示すように、この値を“EDC_VAL”内の第0セクタレジスタに出力する。

【0078】

この後、“ROW”を初期化し（ステップS216）、“COL”を1インクリメントする（ステップS217）。

【0079】

一方、“ROW”が“11”で、“COL”が“42”である場合は、図3に示す“8747”番目のデータ入力に対応した4 Byte入力EDC演算処理が終了した状態であるため、“EDC_VAL”と“TMP_EDC_VAL_1”とのEXORをとった値を再度“EDC_VAL”に保持し（ステップS218）、“ROW”及び“COL”を初期化し（ステップS219、ステップS220）、処理を終了する。

【0080】

以上のステップにより、図3に示すECCブロック内第0セクタのEDC演算結果が全て求まり、処理を終了する。

【0081】

そして、全てのセクタに対する第1のEDC処理が終了した後、前記誤り訂正回路20におけるシンドローム演算値が0でなく、誤りが含まれていると判断された場合は、誤り訂正処理が施される（図1のステップS30）。そして、この誤り訂正処理において得られた誤りデータ位置及び誤りデータ数値を用いて、該誤りデータ位置が示す箇所のための第2のEDC演算処理が行われる。

【0082】

この第2のEDC演算処理については、第2の誤り訂正制御回路142の制御の下、第1のEDC演算処理と同様、演算回路11内の誤り検出符号演算回路111及び誤り検出符号スキップ演算回路112によって、入力される前記データ符号列のうち、前記誤り訂正回路20において得られた誤りデータ位置が示すデータについてのみ、再度EDC演算処理を行うものであり、この演算結果は、図6に示すように、第1のメモリ13内の第2の誤り検出符号演算結果保持部132（以下、“TMP_EDC_VAL_2”と称す。）に保持される。この“TMP_EDC_VAL_2”に保持される値は、前述したように、当該誤り検出装置10より得られる正しいEDC演算結果と、第1のEDC演算処理により得られ、現在“EDC_VAL”に保持されている第1の誤り検出符号演算結果との差分情報となる。

【0083】

従って、前記誤り訂正回路20において得られた前記誤りデータ位置に対して第2のEDC処理が終了後、前記第2の誤り検出制御回路142は、図6に示すように、前記“TMP_EDC_VAL_2”と、前記“EDC_VAL”のEXORをとることによって、前記“EDC_VAL”の値を更新し、正しい誤り検出符号演算結果を得る。

【0084】

ここで、本誤り検出回路10において誤り検出符号演算処理を行う際に、2つのメモリ、“TMP_EDC_VAL”（第1のメモリ13）及び“EDC_VAL”（第2のメモリ12）を使用する理由を以下に示す。

【0085】

図3に示すC2方向のECC処理における第0セクタの“COL”が“1”の列について、データ入力順序は、0番目、1番目、2番目、・・・、11番目以降も、第2セクタである12番目、13

番目、・・・のように引き続いて入力され、図9に示すECCブロック最終セクタである第15セクタ最終行の191番目以降、C2パリティデータである最終行の207番目、続いて1インクリメントされた“COL”について、図3に示す208番目、209番目、・・・といった順序となる。従って、入力されたデータについて、前回までに入力されたデータ符号列の誤り検出符号を元にEDCスキップ演算を行った結果を保持するメモリは、第0セクタから第15セクタまでの全ての位置においても同様に使用されるため、第0セクタから第15セクタまでの任意のセクタにおけるEDC演算結果を保持するメモリが、各セクタ毎で個別に使用される必要がある。よって、本実施の形態1では、第1のメモリ13において各セクタ内の1列分のEDC演算結果を、EXORをとることによって得、この得た第1のメモリ13の値を、第2のメモリ12内に設けられた各セクタ毎のレジスタに保持して、前記第1のメモリ13内の値をリセットし、該第1のメモリ13を次のセクタの1列分のEDC演算結果を得るために使用することができるようにしたものである。

【0086】

次に、前述した誤り検出符号演算回路111におけるEDC演算処理、及び誤り検出符号ステップ演算回路112におけるEDCスキップ演算処理の詳細について、図13の4Byte入力EDC演算回路を用いて説明する。

【0087】

図13に示すように、EDC演算回路は32ビットのシフトレジスタで構成されるため、32ビット全てに0が存在するEDC演算回路に対して、任意のビット値をEDC演算回路に入力した後のEDC演算結果である32ビットのシフトレジスタ値は、32ビット中の各ビットについて、任意のビット値を入力する前の各ビット間の関係式で表現することができる。

【0088】

例えば、0の値である1ビットが図13のEDC演算回路に入力された場合、32ビット中の各ビット値は、0の値である1ビットが入力される前の32ビットそれぞれの状態をBit'として以下のように示すことができる。

【0089】

Bit[0] = Bit' [31]
 Bit[1] = Bit' [0]
 Bit[2] = Bit' [1]
 Bit[3] = Bit' [2]
 Bit[4] = Bit' [3] ^ Bit' [31]
 Bit[5] = Bit' [4]

Bit[30] = Bit' [29]
 Bit[31] = Bit' [30] ^ Bit' [31]

【0090】

前述した1ビットスキップ演算処理式は、前回にEDC演算回路に対して入力されたデータと、新たにEDC演算回路に対して入力されたデータ間の差分が1ビットであるときに用いる、演算回路11内の誤り検出符号演算回路111の演算処理式に他ならない。

【0091】

さらに、図4に示すステップS209他で、データの連続性をたもつために、データを168ByteスキップさせるEDCスキップ演算処理を考えると、32ビット全てに0が存在する図13で示すEDC演算回路に対して、全て0の値である1344ビット(168Byte)が入力された後のEDC演算結果である32ビットのシフトレジスタ値は、0の値である1ビットが入力される前の32ビットそれぞれの状態をBit'として以下のように示すことができる。

【0092】

Bit[0] = Bit' [0] ^ Bit' [1] ^ Bit' [2] ^ Bit' [3] ^ Bit' [5] ^ Bit' [6]
 ^ Bit' [7] ^ Bit' [9] ^ Bit' [10] ^ Bit' [11] ^ Bit' [13] ^ Bit' [14] ^ Bit'
 [15] ^ Bit' [17] ^ Bit' [18] ^ Bit' [19] ^ Bit' [21] ^ Bit' [22] ^ Bit' [23] ^ Bit'

$$\text{Bit}[24] \wedge \text{Bit}'[25] \wedge \text{Bit}'[26] \wedge \text{Bit}'[27] \wedge \text{Bit}'[28] \wedge \text{Bit}'[29] \wedge \text{Bit}'[30] \wedge \text{Bit}'[31]$$

$$\text{Bit}[1] = \text{Bit}'[0] \wedge \text{Bit}'[1] \wedge \text{Bit}'[2] \wedge \text{Bit}'[3] \wedge \text{Bit}'[4] \wedge \text{Bit}'[6] \wedge \text{Bit}'[7] \wedge \text{Bit}'[8] \wedge \text{Bit}'[10] \wedge \text{Bit}'[11] \wedge \text{Bit}'[12] \wedge \text{Bit}'[14] \wedge \text{Bit}'[15] \wedge \text{Bit}'[16] \wedge \text{Bit}'[18] \wedge \text{Bit}'[19] \wedge \text{Bit}'[20] \wedge \text{Bit}'[22] \wedge \text{Bit}'[23] \wedge \text{Bit}'[24] \wedge \text{Bit}'[25] \wedge \text{Bit}'[26] \wedge \text{Bit}'[27] \wedge \text{Bit}'[28] \wedge \text{Bit}'[29] \wedge \text{Bit}'[30] \wedge \text{Bit}'[31]$$

$$\text{Bit}[2] = \text{Bit}'[0] \wedge \text{Bit}'[1] \wedge \text{Bit}'[2] \wedge \text{Bit}'[3] \wedge \text{Bit}'[4] \wedge \text{Bit}'[5] \wedge \text{Bit}'[7] \wedge \text{Bit}'[8] \wedge \text{Bit}'[9] \wedge \text{Bit}'[11] \wedge \text{Bit}'[12] \wedge \text{Bit}'[13] \wedge \text{Bit}'[15] \wedge \text{Bit}'[16] \wedge \text{Bit}'[17] \wedge \text{Bit}'[19] \wedge \text{Bit}'[20] \wedge \text{Bit}'[21] \wedge \text{Bit}'[23] \wedge \text{Bit}'[24] \wedge \text{Bit}'[25] \wedge \text{Bit}'[26] \wedge \text{Bit}'[27] \wedge \text{Bit}'[28] \wedge \text{Bit}'[29] \wedge \text{Bit}'[30] \wedge \text{Bit}'[31]$$

$$\text{Bit}[3] = \text{Bit}'[0] \wedge \text{Bit}'[1] \wedge \text{Bit}'[2] \wedge \text{Bit}'[3] \wedge \text{Bit}'[4] \wedge \text{Bit}'[5] \wedge \text{Bit}'[6] \wedge \text{Bit}'[8] \wedge \text{Bit}'[9] \wedge \text{Bit}'[10] \wedge \text{Bit}'[12] \wedge \text{Bit}'[13] \wedge \text{Bit}'[14] \wedge \text{Bit}'[16] \wedge \text{Bit}'[17] \wedge \text{Bit}'[18] \wedge \text{Bit}'[20] \wedge \text{Bit}'[21] \wedge \text{Bit}'[22] \wedge \text{Bit}'[24] \wedge \text{Bit}'[25] \wedge \text{Bit}'[26] \wedge \text{Bit}'[27] \wedge \text{Bit}'[28] \wedge \text{Bit}'[29] \wedge \text{Bit}'[30] \wedge \text{Bit}'[31]$$

$$\text{Bit}[4] = \text{Bit}'[4] \wedge \text{Bit}'[24]$$

$$\text{Bit}[5] = \text{Bit}'[5] \wedge \text{Bit}'[25]$$

$$\text{Bit}[30] = \text{Bit}'[2] \wedge \text{Bit}'[3] \wedge \text{Bit}'[6] \wedge \text{Bit}'[7] \wedge \text{Bit}'[10] \wedge \text{Bit}'[11] \wedge \text{Bit}'[14] \wedge \text{Bit}'[15] \wedge \text{Bit}'[18] \wedge \text{Bit}'[19] \wedge \text{Bit}'[30]$$

$$\text{Bit}[31] = \text{Bit}'[0] \wedge \text{Bit}'[1] \wedge \text{Bit}'[2] \wedge \text{Bit}'[4] \wedge \text{Bit}'[5] \wedge \text{Bit}'[6] \wedge \text{Bit}'[8] \wedge \text{Bit}'[9] \wedge \text{Bit}'[10] \wedge \text{Bit}'[12] \wedge \text{Bit}'[13] \wedge \text{Bit}'[14] \wedge \text{Bit}'[16] \wedge \text{Bit}'[17] \wedge \text{Bit}'[18] \wedge \text{Bit}'[20] \wedge \text{Bit}'[21] \wedge \text{Bit}'[22] \wedge \text{Bit}'[23] \wedge \text{Bit}'[24] \wedge \text{Bit}'[25] \wedge \text{Bit}'[26] \wedge \text{Bit}'[27] \wedge \text{Bit}'[28] \wedge \text{Bit}'[29] \wedge \text{Bit}'[30]$$

【0093】

前述した168Byteスキップ演算処理式は、前回にEDC演算回路に対して入力されたデータと、新たにEDC演算回路に対して入力されたデータ間の差分が168Byteであるときに用いる、演算回路11内の誤り検出符号スキップ演算回路112の1つの演算処理式に他ならない。そして、図4のステップS213においては、各“COL”値に応じたスキップ演算処理を行う必要があり、この各“COL”値に応じたスキップ演算処理式についても、前記168Byteスキップ演算処理式と同様にして得ることができ、図4におけるステップS213では、各“COL”値に応じたスキップ演算処理式を42通り用意すればよいこととなる。

【0094】

なお、前述したように図4におけるステップS213の処理のために、“COL”値に応じたスキップ演算処理式を全部で42通り用意するのではなく、前記スキップ演算を分割して処理するようにすれば、前記ステップS213におけるステップ数を削減することが可能となる。

【0095】

以下、スキップ演算処理を分割処理により行う場合について、図7の168ByteスキップEDC補正演算処理説明図、及び図8の168ByteスキップEDC補正演算処理の分割処理説明図を用いて説明する。

【0096】

図7に示すように、図3に示す“COL”が“0”の場合は、“ROW”が“11”の時点で168ByteスキップさせるEDCスキップ演算を行う必要がある。このように、セクタ中の全てのEDC演算を行うためには、EDCスキップ演算処理は“COL”が“0”から“41”までの42通りについて用意しておく必要があり、処理資源を大きく消費してしまう。従って、例えば、データを168ByteスキップさせるEDCスキップ演算処理を行うことと、42Byteスキップさせ

るEDCスキップ演算処理を4回行うことが等価になるので、このことを利用して、データを168ByteスキップさせるEDCスキップ演算処理を個別に用意するのではなく、別個に存在する、データを42ByteスキップさせるEDCスキップ演算処理を4回再利用する方法に変更する。

【0097】

以上のように、本実施の形態1によれば、入力されたデータ符号列に対して、入力データ単位Byteに対応した第1のEDC演算処理を行い、その演算結果を第1のメモリ13内の第1の誤り検出符号演算結果保持部131に保持した後に、該第1の誤り検出符号演算結果保持部131の値と、次に入力されるデータまでの差分Byte数のEDCスキップ演算処理の演算結果のEXORをとることを繰り返すようにしたので、EDC演算処理を行う際に、入力されるデータが論理的な連続性のない方向(図9のC2方向)であっても、EDC演算処理を実行することが可能となり、前記ECC及びEDCの両処理の半導体集積回路の実装を実現できる。また、これにより、ECC処理とEDC処理を同時実施が可能となるので、ECCブロックをバッファから一度読み出すだけで両処理が行え、ECC処理のためにバッファから読み出した1ECCブロックを、ECC処理後に再度読み出してEDC演算処理を行う必要がなくなり、メモリバッファのバンド幅の消費、及び処理時の肥大化を低減することができる。

【0098】

また、前記演算回路11内の誤り検出符号スキップ演算回路112における、EDCスキップ演算処理を、分割処理方法を用いて行えば、セクタ内のEDC処理単位列に対応した、全てのEDCスキップ演算処理を用意することなく、別個に存在するEDCスキップ演算処理を複数回再利用することができ、この結果、処理資源の消費を抑えることが可能となる。

【0099】

なお、前述の説明においては、データに論理的な連続性のないC2方向のECC処理とEDC演算処理とを同時実行する場合のEDC演算処理について詳述したが、本実施の形態1にかかる誤り検出回路10において、データに論理的な連続性があるC1方向のECC処理とEDC演算処理とを同時実行することも可能である。この場合第1のEDC演算処理は、前述したC2方向の処理とは異なりスキップ演算をする必要がないので、演算回路11内の誤り検出符号演算回路111のみを使用し、演算処理単位(ここでは4Byte)毎に得られた演算結果を第1のメモリ13内の第1の誤り検出符号演算結果保持部131に保持していき、1セクタのEDC演算処理が終了後、前記第1の誤り検出符号演算結果保持部131に保持した演算結果を、第2のメモリ12内の対応するセクタレジスタに保持し、次のセクタのEDC演算処理に移行する処理を繰り返す。そして、第2のEDC演算処理は、前述したC2方向の処理と同様、前記誤り検出回路20にて得られた誤りデータ位置及び誤りデータ数値を元に、入力されるデータ符号列のうち、前記誤りデータ位置のデータに対してのみ再度EDC演算処理を行い、得られた演算結果を第1のメモリ13内の第2の誤り検出符号演算結果保持部132に保持し、該第2のEDC処理が終了後、その時点で第2のメモリ12に保持されている第1のEDC演算結果を、前記第2の誤り検出符号演算結果保持部132に保持された演算結果を用いて更新し、正しいEDC演算結果を得る。

【産業上の利用可能性】

【0100】

本発明の誤り検出装置及び誤り検出方法は、ECC復号化されたデジタルデータを高倍速で記録または再生する光ディスク記録再生装置において、データを記録再生するときに、誤り訂正及び誤り検出を同時処理させる際に有用である。

【図面の簡単な説明】

【0101】

【図1】本発明の実施の形態1における、ECC処理とEDC処理の同時実行した際のパイプライン処理を示す図である。

【図2】本発明の実施の形態1における、誤り検出回路の構成を示す図である。

【図3】本発明の実施の形態1における、C2方向訂正時に第0セクタにおいてデータ入力単位を4Byteとしたときの誤り訂正回路、及び誤り検出回路に対するデータ入力

順序を示す図である。

【図 4】本発明の実施の形態 1 における誤り検出回路において、C2 方向訂正時の第 1 の EDC 演算処理の一連の流れを示すフローチャート図である。

【図 5】本発明の実施の形態 1 における誤り検出回路の C2 方向訂正時における第 1 の EDC 演算処理のデータの流れを示す図である。

【図 6】本発明の実施の形態 1 における誤り検出回路の、C2 方向訂正時における第 2 の EDC 演算処理のデータの流れを示す図である。

【図 7】本発明の実施の形態 1 における誤り検出回路の、誤り検出符号スキップ演算回路において、データを 168Byte スキップさせる EDC スキップ演算処理を示す図である。

【図 8】本発明の実施の形態 1 における誤り検出回路の、誤り検出符号スキップ演算回路において、データを 168Byte スキップさせる EDC スキップ演算処理の分割処理を示す図である。

【図 9】DVD に記録されたデータを誤り訂正単位ブロック (ECC ブロック) に分けたときの ECC ブロック構成図である。

【図 10】ECC ブロックの C1 方向についての誤り訂正実施例を示す図である。

【図 11】ECC ブロックの C2 方向についての誤り訂正実施例を示す図である。

【図 12】ECC ブロック上のセクタの構成を示す図である。

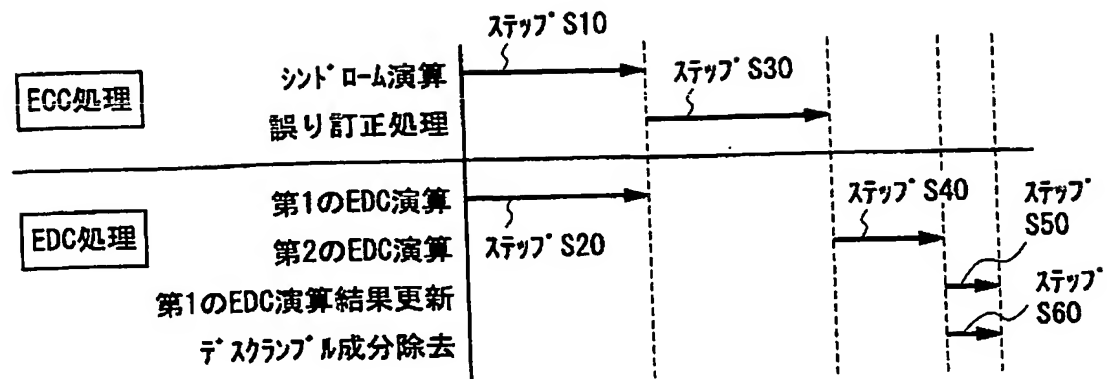
【図 13】4 Byte 入力の EDC 演算回路を示す図である。

【符号の説明】

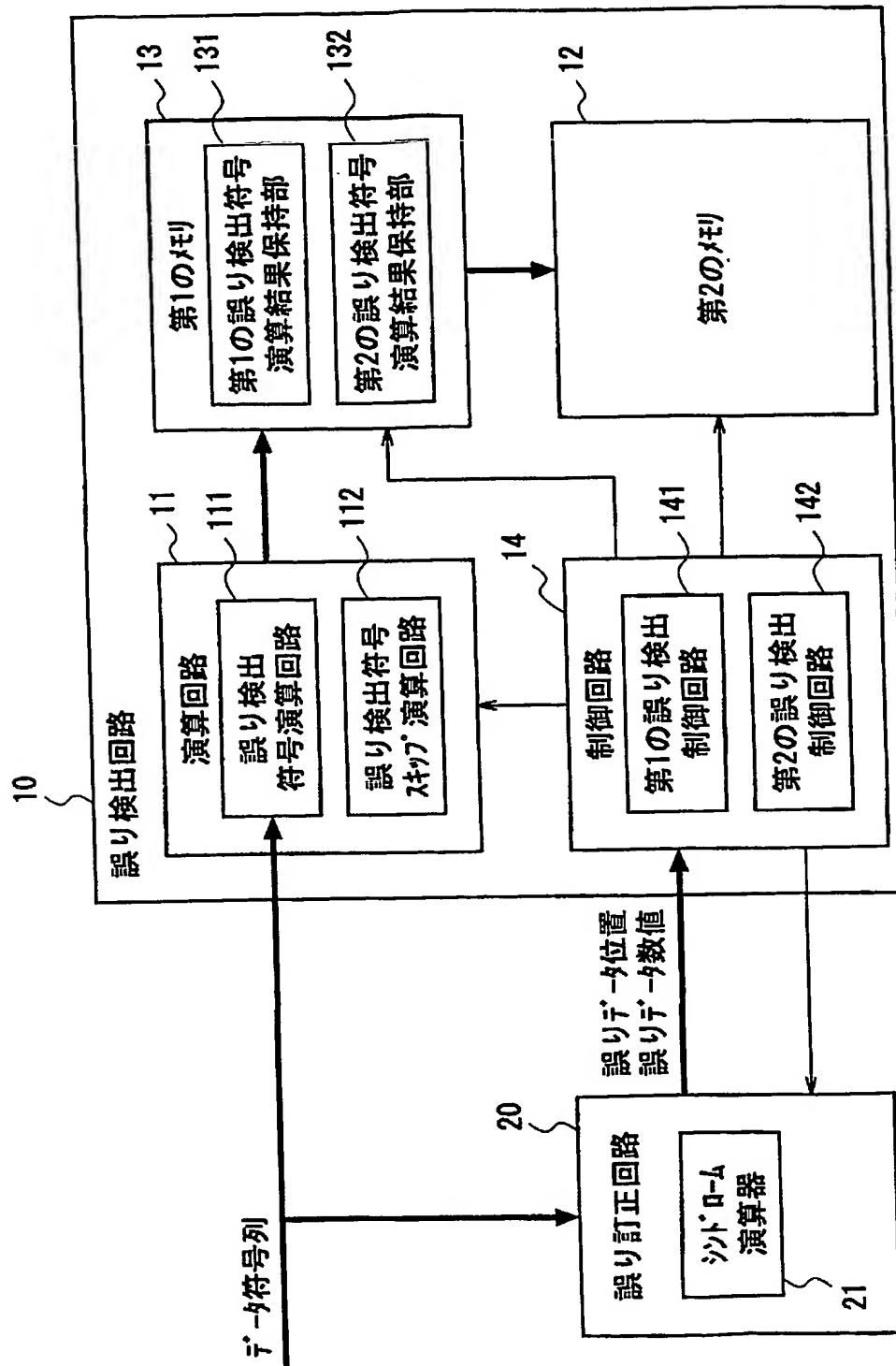
【0102】

- 10 誤り検出回路
- 11 演算回路
- 12 第 2 のメモリ
- 13 第 1 のメモリ
- 14 制御回路
- 20 誤り訂正回路
- 21 シンドローム演算器
- 111 誤り検出符号演算回路
- 112 誤り検出符号スキップ演算回路
- 131 第 1 の誤り検出符号演算結果保持部
- 132 第 2 の誤り検出符号演算結果保持部
- 141 第 1 の誤り制御回路
- 142 第 2 の誤り制御回路

【書類名】 図面
【図 1】



【図2】



【図 3】

データ
入力順序

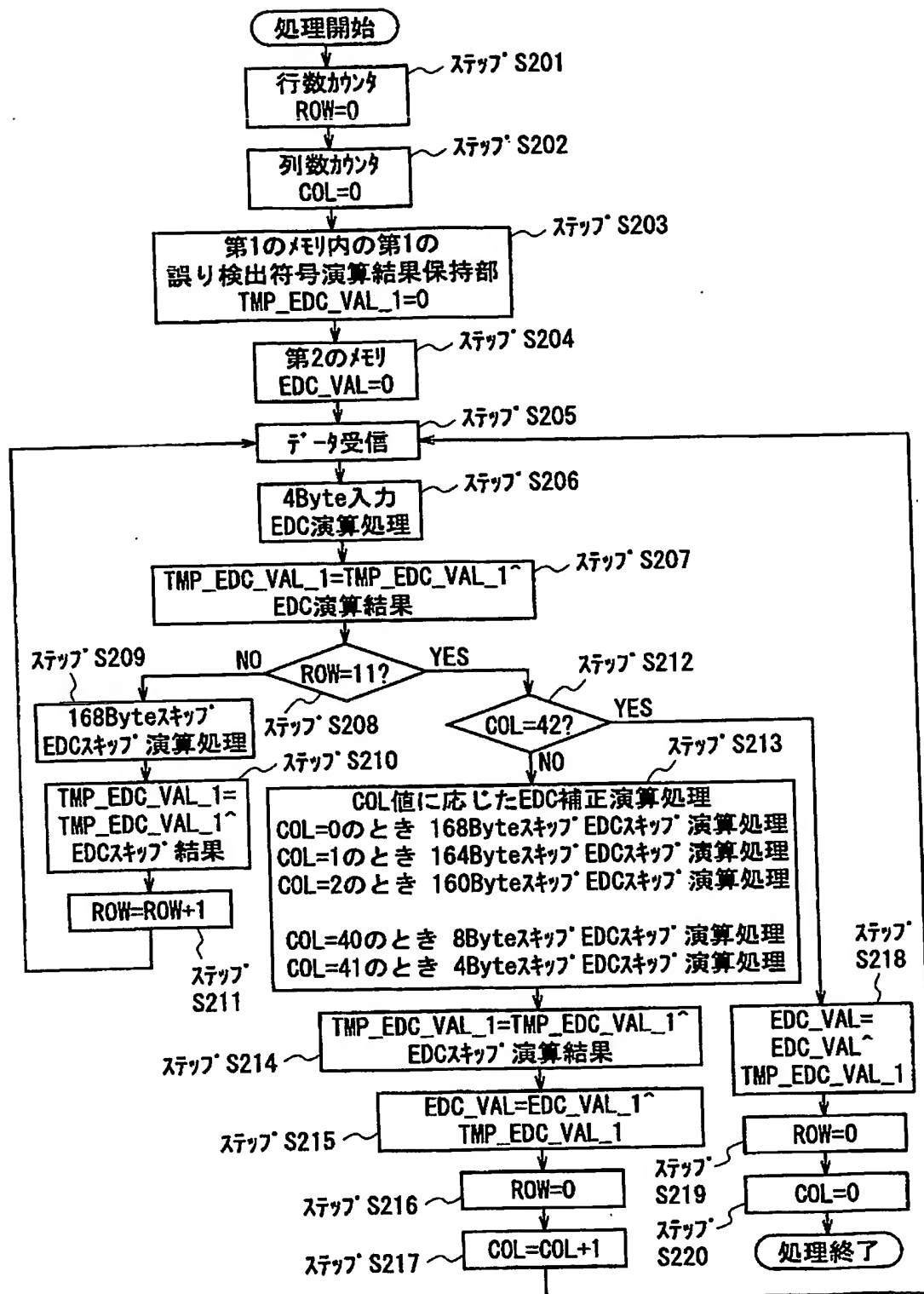
→ EDC演算順序

0	208	8528	8736	ROW=0
1	209	8529	8737	ROW=1
2	210	8530	8738	ROW=2
3	211	8531	8739	ROW=3
4	212	8532	8740	ROW=4
5	213	8533	8741	ROW=5
6	214	8534	8742	ROW=6
7	215	8535	8743	ROW=7
8	216	8536	8744	ROW=8
9	217	8537	8745	ROW=9
10	218	8538	8746	ROW=10
11	219	8539	8747	ROW=11

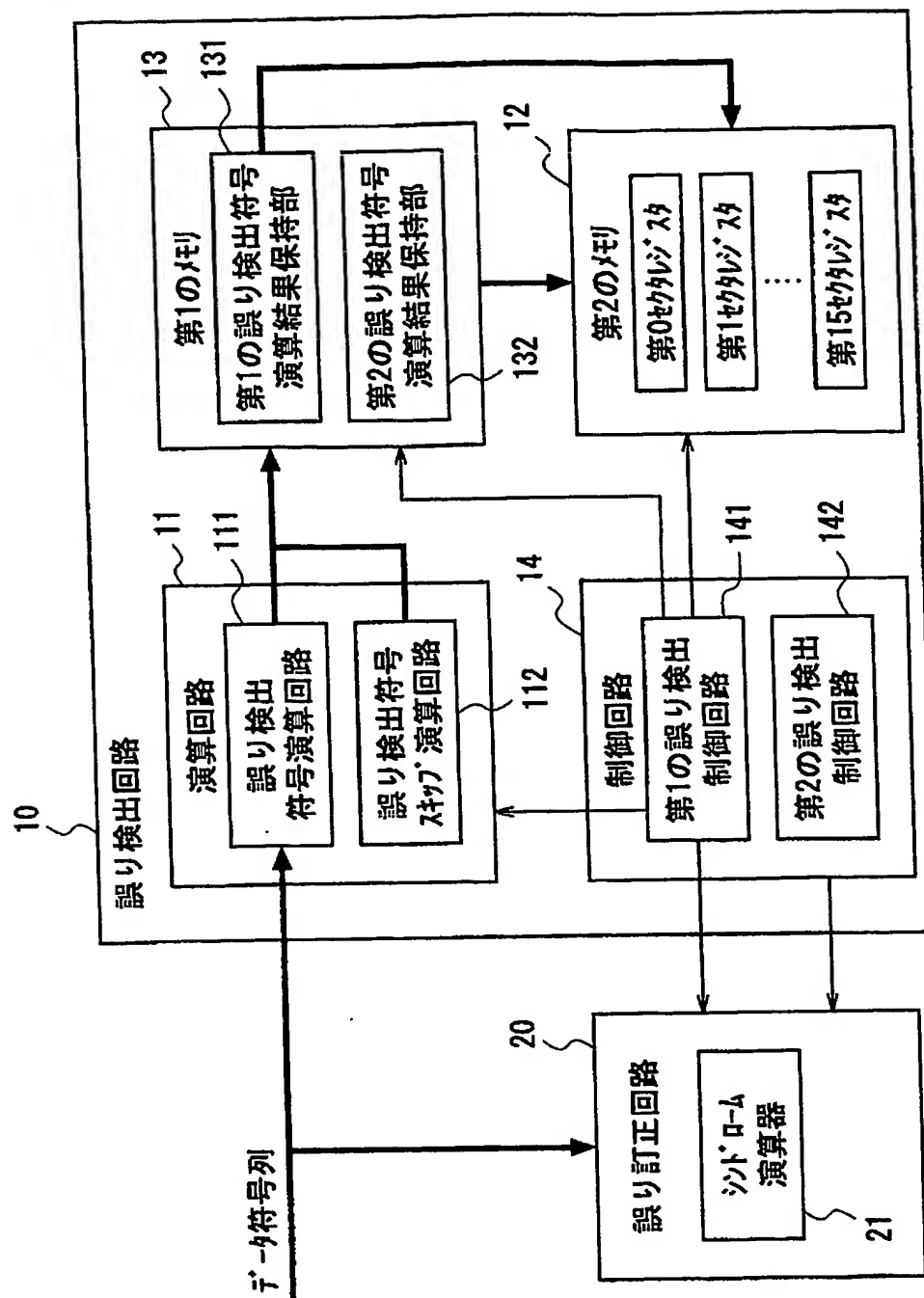
COL=0 COL=1 COL=2 COL=3

COL=41 COL=42

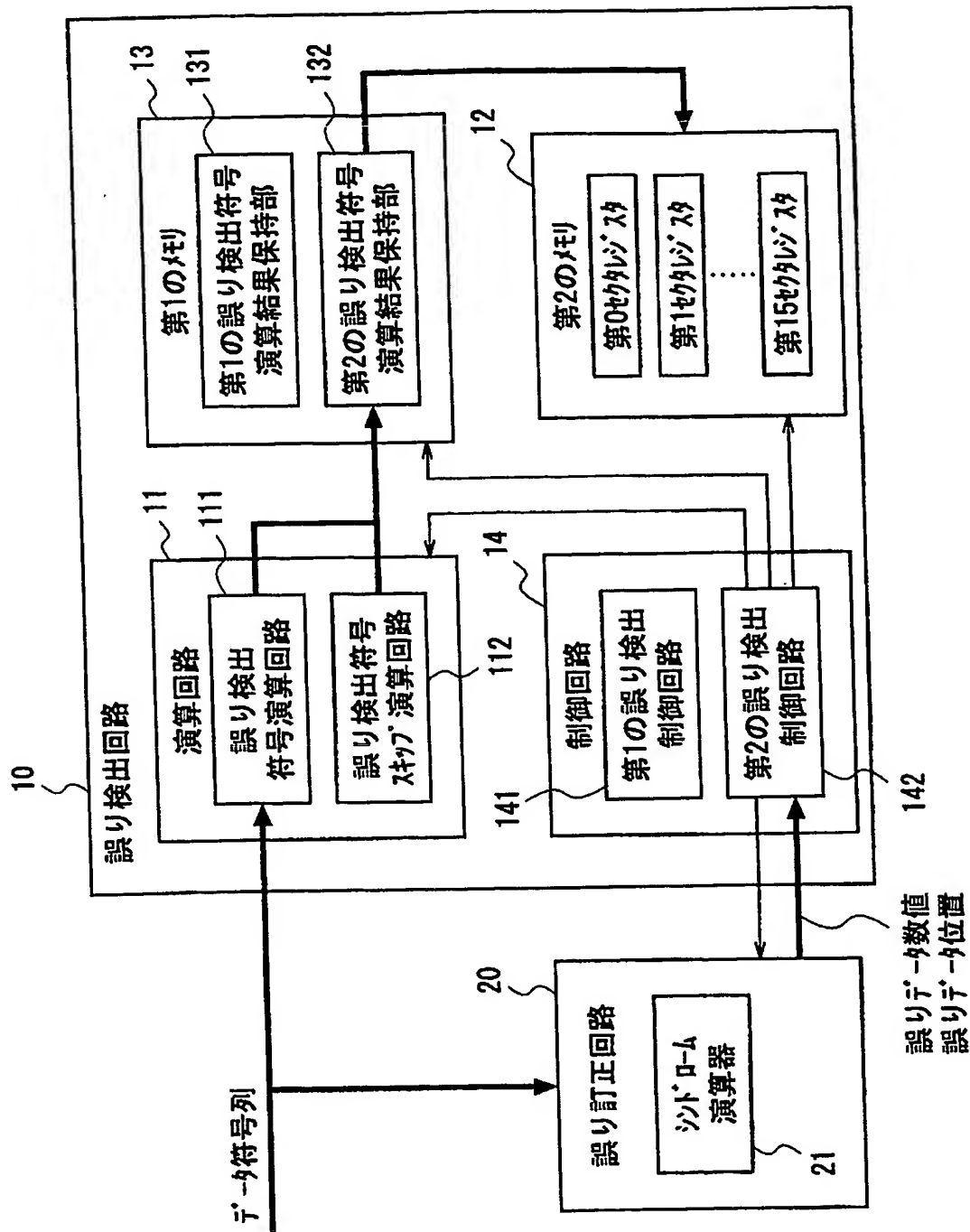
【図 4】



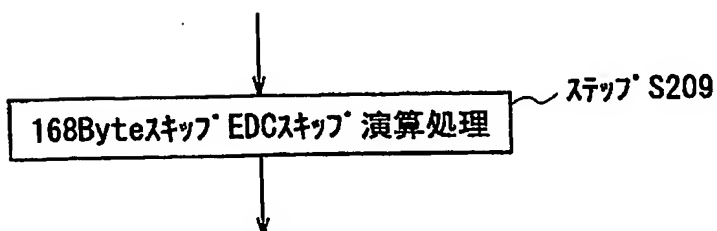
【図5】



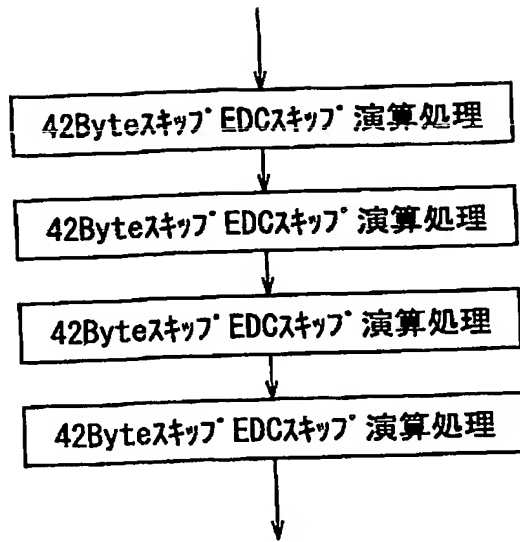
【図 6】



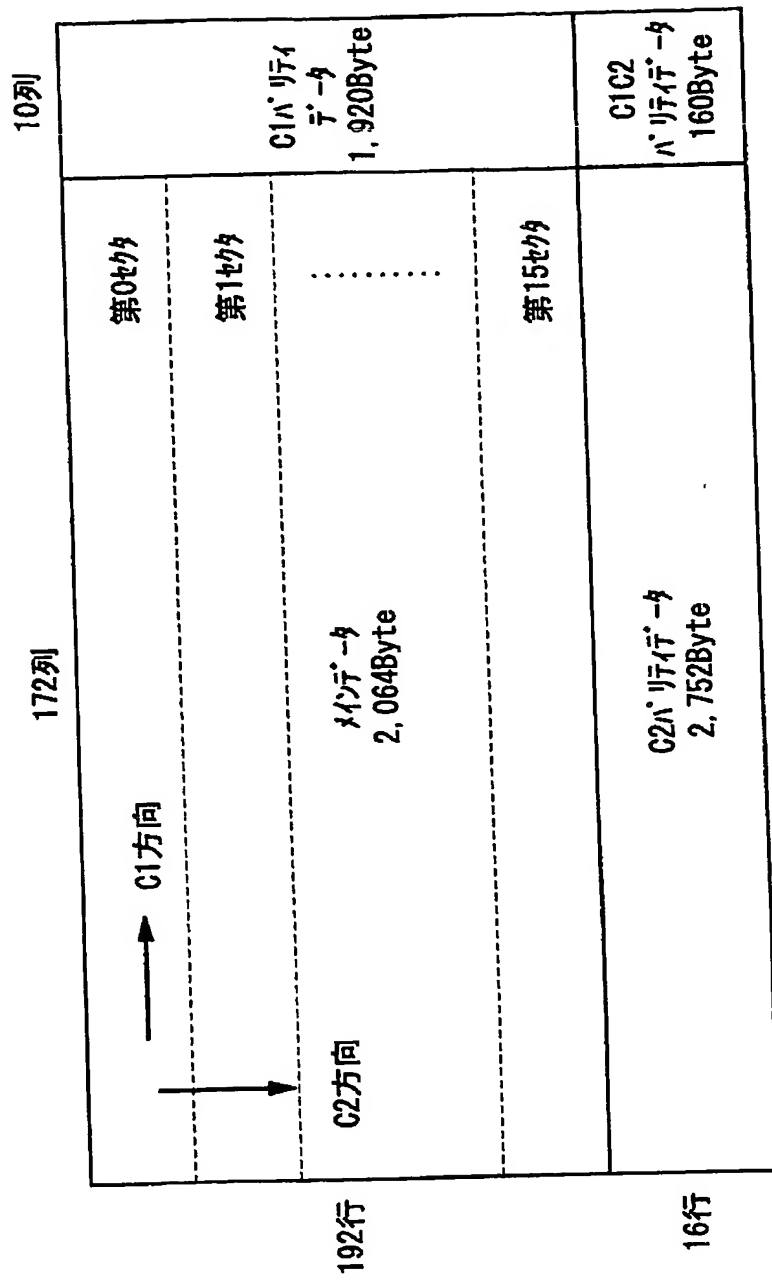
【図 7】



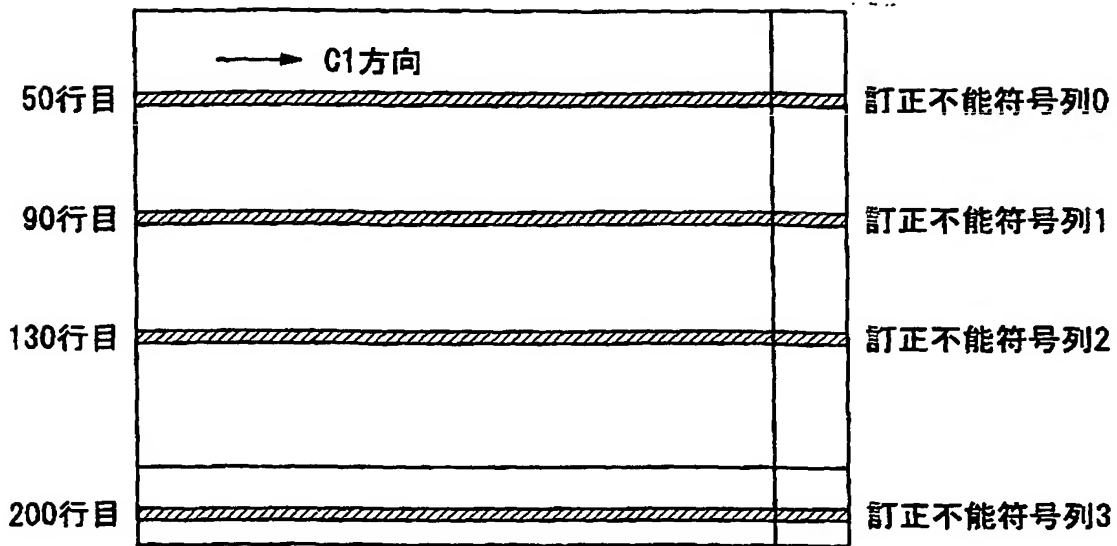
【図 8】



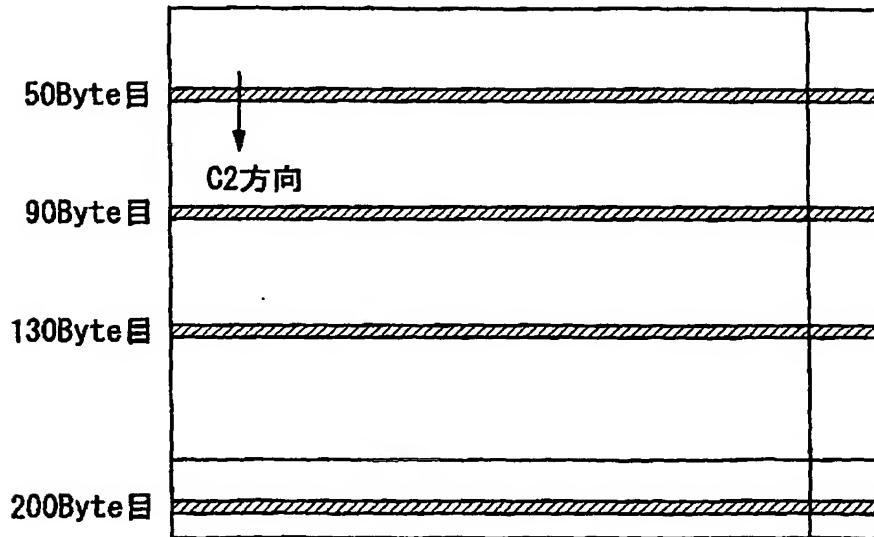
【図 9】



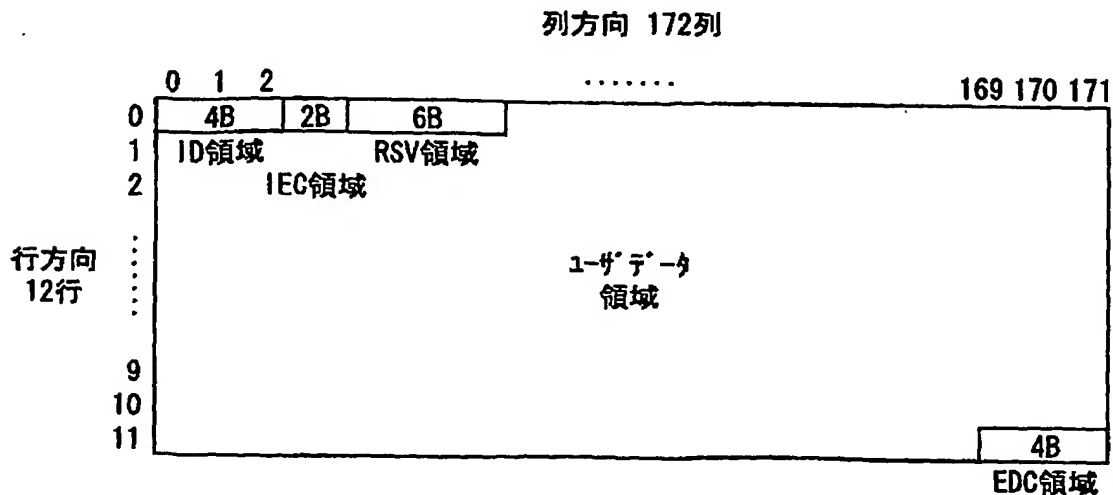
【図10】



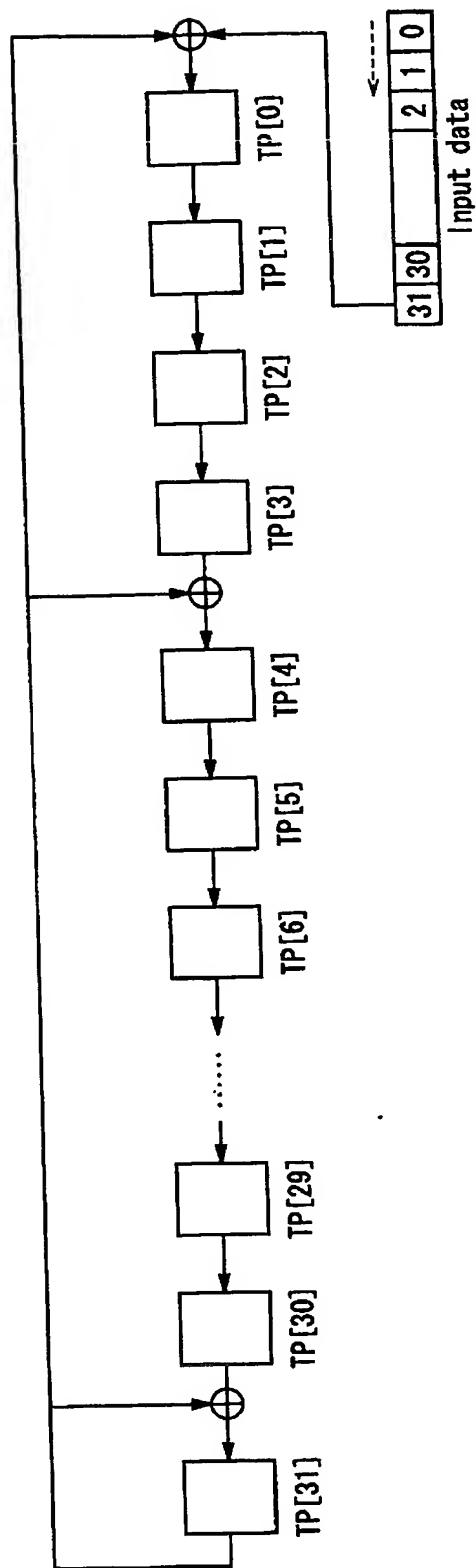
【図11】



【図12】



【図 13】



【書類名】 要約書

【要約】

【課題】 ECCブロックの誤り検出を行う対象符号列が、論理的に連続的でない並びで入力された際にも、誤り訂正と誤り検出を同時に行える誤り検出装置及び誤り検出方法を提供する。

【解決手段】 誤り訂正と同時に誤り検出を行う誤り検出装置であって、ECCブロックをバッファから1度だけ読み出し、誤り検出において、該誤り検出を行う対象符号列が論理的に連続的でない並びで入力された際には、該符号列に連続性をもたせるスキップ演算処理を行うことにより、論理的に連続的でない並びで入力されたECC処理とEDC演算処理とを同時に行うことを可能にする。

【選択図】 図1

特願 2 0 0 3 - 3 0 7 5 0 9

出 願 人 履 歴 情 報

識別番号

[0 0 0 0 0 5 8 2 1]

1. 変更年月日

1 9 9 0 年 8 月 2 8 日

[変更理由]

新規登録

住 所

大阪府門真市大字門真 1 0 0 6 番地

氏 名

松下電器産業株式会社